Release Notes for Apex version 2.4

Note:

- These release notes may be revised after the release of Apex 2.4. Check the Apex web site (http://human-factors.arc.nasa.gov/apex) for updates.

- Apex 2.4 differs from previous versions in how applications (simworlds) are defined and structured. Applications written for previous versions need to be modified to work in 2.4; instructions on how to do this are given in the *Upgrading existing applications* section below. Chapter 3 of the Apex Reference Manual explains these and related changes.

Enhancements

- Multi-agent support - Apex can now control/simulate more than one agent at a time: agents can use separate PDL procedure sets; agents can have different top-level tasks (`do-domain` is still the default); agents can interact with one another as first-class simulation objects.

- Improvements to how application (model) code is organized and specified: no restriction on names of application files (e.g. no more initialize.lisp, etc…); the entire application (including PDL) can be in a single file if desired; native applications and foreign applications (e.g. X-Plane) are specified uniformly; support is provided for PDL and Lisp code libraries; support is provided for tracking the Apex version compatibility of applications.

- Interface improvements: trace can filtered to include only events involving a specified object by clicking on that object in the hierarchical view; select an object and all its components by clicking the object-select button; click on agent to restrict trace, PERT and preview views to the selected agent; shift-click to select multiple agents; click once on object in any view to inspect; inspect any object by typing its name into inspect text field; load application (native or foreign) from file browser or from list of recently loaded applications; application can be paused.

- Example applications: Kitchenworld now works correctly; new introductory application Hello World; new application Roshambo included to illustrate multiagent capability; X-Plane application code streamlined, improved as example of how to construct foreign applications; other applications restructured to take advantage of library support

- Miscellaneous: New documentation added to Apex manual (but see Known Problems below); user settings file supports customizing Apex behavior.

Known Problems

- The Apex Reference Manual is not complete. It will continue to be revised, and the Apex web site (http://human-factors.arc.nasa.gov/apex) should be checked for updates. In particular, Chapter 1 has not been updated to point out Apex's increasing role as a general autonomy tool (e.g. though non-native application support explained in Chapter 3). Chapter 5 is just a beginning of a much-needed Apex programming reference. Please inform us of any missing or incorrect information in the manual.

- When using Apex with Sherpa, an error in Apex (Lisp) is likely to cause Apex and Sherpa to become out of sync and necessitates resetting Sherpa (select Reset from its File menu).

- Using a string as an agent name causes traces to stop appearing in Sherpa. For now use symbols instead.

- In Sherpa, the object tree sometimes does not appear on initial loading of the application.

- If a non-native application is loaded and run, followed by a native application, and followed again by a non-native application, the time gets stuck at 0.

- When loading an application, Sherpa and Apex are unable to distinguish an application definition file (ADF) from any other kind of file. Unspecified results will occur if a non-ADF is loaded.

- When Sherpa and Apex are run on different machines, new applications cannot be loaded by Sherpa. They need to be loaded in the Listener, after which they will become recent applications selectable in Sherpa (see Chapter 3 of the Apex Reference Manual).

- Worldbuilder is in the process of being upgraded to be compatible with Apex 2.4, but is not ready yet. If you use Worldbuilder, you will have to hand-modify its output as specified in the Upgrading Existing Applications section below.

- An error appears when the `start-activity` task is attempting to control a resource. The message says that the task does not have ownership. A workaround is to add an extra level and have the initial task call the procedure that uses the resource.

- A bus error in Lisp appears when a typo in an Apex application, `:initargs` instead of `:initarg`, is used.

Upgrading Existing Applications

Perform the following modifications to pre-2.4 applications to make them 2.4 compliant.

1. Add an `apex-info` form (Chapter 3) to the top of all application files (e.g. pdl.lisp, definitions.lisp).

2. Create an application definition file and put a `defapplication` form in it. For example, let's assume your simworld is called my-world and has three files: `initialize.lisp`, `definitions.lisp`, and `pdl.lisp`. Create the file `myworld.lisp` with the following contents:

   ```
   (apex-info :version "2.4")

   (defapplication "My World"
      :libraries ()
      :files (definitions pdl initialize)
      :init-sim (initialize-my-world))
   ```

   You'll need to edit the `defapplication` form as explained the following steps.

3. In `initialize.lisp`, change (`initialize-simulation` ...) to:

   ```
   (defun initialize-my-world ()
       ...)
   ```

4. Read about libraries in Chapter 3. Look at the contents of `apex2.4b/apexlib` and add any libraries needed by your application to the list following `:libraries` in `defapplication`. Most likely, the needed entry will be:

   ```
   :libraries ("human")
   ```

5. Remove the `:cause` keyword parameters from all functions. The causal tracing mechanism (which was never documented) is being revamped and this parameter is no longer supported.

6. If you use the macros `class-maker`, `instance-maker`, or `action-maker`, please note that these are deprecated, though will remain supported indefinitely via a new user library "maker". Add this to the `:libraries` list exemplified above. Eventually, transform this code by calling the Lisp function `macroexpand` on the forms and substituting the resulting CLOS forms in your file.

7. The `world-event` activity has been renamed to `external-event`; `dummy-act` and several other activities whose name ended in `"-act"` have been obsoleted and replaced by `resource-activity`; the PDL procedure `stop` has been renamed to `end-trial`. Edit your application to perform these translations. Hint: one way to find all the `"-act"` activities that need translation is to run your application and see where it breaks.

8. The above steps are just one way of transforming an existing application. In particular, there are no longer any file naming requirements, so the application may be restructured in any way desired, including being made into one file. The `defapplication` form need not be placed in its own file.